
Subject: Re: [PATCH net-2.6.25 1/3] Uninline the __inet_hash function

Posted by Eric Dumazet on Wed, 19 Dec 2007 12:21:37 GMT

[View Forum Message](#) <> [Reply to Message](#)

> This one is used in quite many places in the networking code and
> seems to big to be inline.

>

> After the patch net/ipv4/build-in.o loses 725 bytes:

> add/remove: 1/0 grow/shrink: 0/5 up/down: 374/-1099 (-725)

> function	old	new	delta
> __inet_hash	-	374	+374
> tcp_sacktag_write_queue	2255	2254	-1
> __inet_lookup_listener	284	274	-10
> tcp_v4_syn_recv_sock	755	495	-260
> tcp_v4_hash	389	40	-349
> inet_hash_connect	1165	686	-479

>

> Exporting this is for dccp module.

>

> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

>

> ---

>

> include/net/inet_hashtables.h | 27 +-----

> net/ipv4/inet_hashtables.c | 27 ++++++-----

> 2 files changed, 29 insertions(+), 25 deletions(-)

>

> diff --git a/include/net/inet_hashtables.h b/include/net/inet_hashtables.h

> index 37f6cb1..1a43125 100644

> --- a/include/net/inet_hashtables.h

> +++ b/include/net/inet_hashtables.h

> @@ -264,31 +264,8 @@ static inline void inet_listen_unlock(struct inet_hashinfo *hashinfo)

> wake_up(&hashinfo->lhash_wait);

> }

>

> -static inline void __inet_hash(struct inet_hashinfo *hashinfo,

> - struct sock *sk, const int listen_possible)

> -{

> - struct hlist_head *list;

> - rwlock_t *lock;

> -

> - BUG_TRAP(sk_unhashed(sk));

> - if (listen_possible && sk->sk_state == TCP_LISTEN) {

> - list = &hashinfo->listening_hash[inet_sk_listen_hashfn(sk)];

> - lock = &hashinfo->lhash_lock;

> - inet_listen_wlock(hashinfo);

> - } else {

```

> - struct inet_ehash_bucket *head;
> - sk->sk_hash = inet_sk_ehashfn(sk);
> - head = inet_ehash_bucket(hashinfo, sk->sk_hash);
> - list = &head->chain;
> - lock = inet_ehash_lockp(hashinfo, sk->sk_hash);
> - write_lock(lock);
> -
> - __sk_add_node(sk, list);
> - sock_prot_inc_use(sk->sk_prot);
> - write_unlock(lock);
> - if (listen_possible && sk->sk_state == TCP_LISTEN)
> - wake_up(&hashinfo->lhash_wait);
> -
> +extern void __inet_hash(struct inet_hashinfo *hashinfo, struct sock *sk,
> + const int listen_possible);
>
> static inline void inet_hash(struct inet_hashinfo *hashinfo, struct sock *sk)
> {
> diff --git a/net/ipv4/inet_hashtables.c b/net/ipv4/inet_hashtables.c
> index 67704da..46f899b 100644
> --- a/net/ipv4/inet_hashtables.c
> +++ b/net/ipv4/inet_hashtables.c
> @@ -267,6 +267,33 @@ static inline u32 inet_sk_port_offset(const struct sock *sk)
>     inet->dport);
> }
>
> +void __inet_hash(struct inet_hashinfo *hashinfo, struct sock *sk,
> + const int listen_possible)
> +{
> + struct hlist_head *list;
> + rwlock_t *lock;
> +
> + BUG_TRAP(sk_unhashed(sk));
> + if (listen_possible && sk->sk_state == TCP_LISTEN) {
> + list = &hashinfo->listening_hash[inet_sk_listen_hashfn(sk)];
> + lock = &hashinfo->lhash_lock;
> + inet_listen_wlock(hashinfo);
> + } else {
> + struct inet_ehash_bucket *head;
> + sk->sk_hash = inet_sk_ehashfn(sk);
> + head = inet_ehash_bucket(hashinfo, sk->sk_hash);
> + list = &head->chain;
> + lock = inet_ehash_lockp(hashinfo, sk->sk_hash);
> + write_lock(lock);
> +
> + __sk_add_node(sk, list);
> + sock_prot_inc_use(sk->sk_prot);
> + write_unlock(lock);

```

```
> + if (listen_possible && sk->sk_state == TCP_LISTEN)
> +   wake_up(&hashinfo->lhash_wait);
> +}
> +EXPORT_SYMBOL_GPL(__inet_hash);
> +
> /*
> * Bind a port for a connect operation and hash it.
> */
```

If you un-inline this (good idea), I am not sure we still need listen_possible argument.

It was usefull only to help compiler to zap dead code (since it was known at compile time), now it only adds some extra test and argument passing.

Thank you
